Resource Selection Using Execution and Queue Wait Time Predictions

Warren Smith

Parkson Wong

Computer Sciences Corporation
NASA Ames Research Center

Motivation

- Grids have lots of different computers
- Where should a user submit their application?
  - Which machines can user access?
  - Which machines have sufficient resources?
  - How much do machines cost to use?
  - When will the application finish?
    - Time to pre-stage files
    - Time waiting in queue
    - Time to execute
    - Time to post-stage files

Approach

- Develop execution time prediction techniques
  - Historical information
  - Instance based learning
- Develop queue wait time prediction techniques
  - Simulate scheduling algorithms
  - Use execution time predictions
- Add them to get turn-around time
- Implement for use at NAS
- Extend to grids

Instance-Based Learning

- Maintain a database of experiences
  - Each experience has a set of input and output features
- Calculate an estimate for a query using relevant experiences
  - Relevance measured with a distance function
  - Calculation can be an average, distance weighted average, locally weighted regression
  - Can use nearest experiences (nearest neighbors) or all
  - Predictions include confidence intervals
- Local learning: don't try to derive one equation that fits all data points
- No learning phase like in neural networks

Distance Functions

- Minkowski
  - Manhattan $D(x,y)=\sum_f |x_f - y_f|$
  - Euclidean $D(x,y)=\sqrt{\sum_f (x_f - y_f)^2}$
  - Only works where the features are linear

$$D(x,y)=\left(\sum_f |x_f - y_f|^r\right)^{1/r}$$

- Heterogeneous Euclidean Overlap Metric
  - Handles features that are linear or nominal

$$d_f(x,y)=\begin{cases}1, & \text{if } x_f \text{ or } y_f \text{ is unknown,}\\ overlap_f(x,y), & \text{if } f \text{ is nominal,}\\ rn\_diff_f(x,y), & \text{otherwise}\end{cases}$$

$$overlap_f(x,y)=\begin{cases}0, \text{ if } x_f = y_f\\ 1, \text{ otherwise}\end{cases}$$

$$D(x,y)=\sqrt{\sum_f d_f(x,y)^2}$$

$$rn\_diff_f(x,y)=\frac{|x_f - y_f|}{\max_f - \min_f}$$

Kernel Regression

- Estimate is weighted average of experiences based on distance
- Weighting is also called kernel function

$$E_f(q) = \frac{\sum_e K(D(q,e))V_f(e)}{\sum_e K(D(q,e))}$$

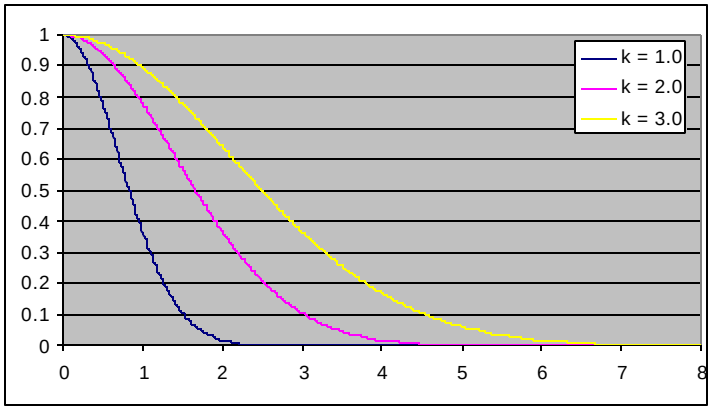
- Want weight->C as d->0 and weight->0 as d->∞
- Gaussian is an example:

$$K(d) = e^{-d^2}$$

- Kernel width k to scale distances:

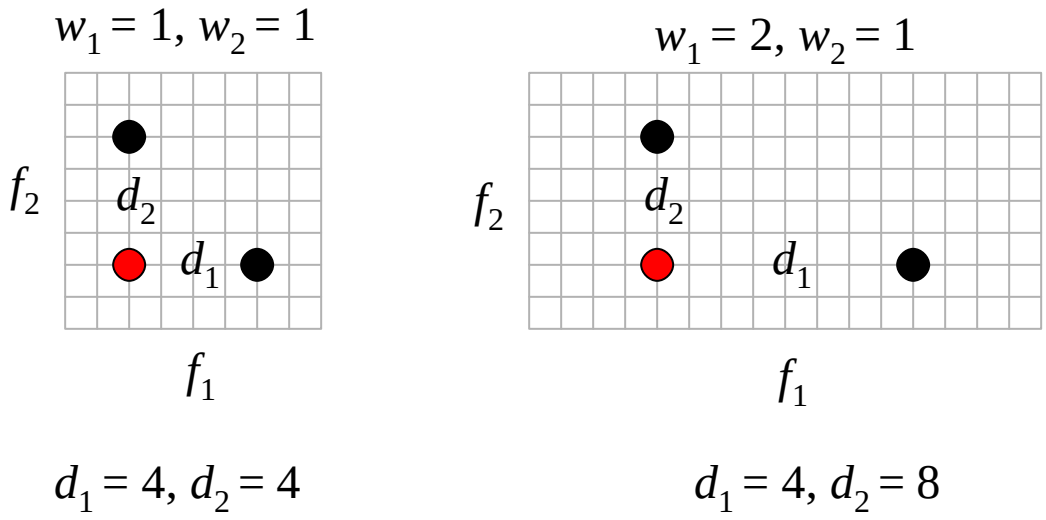
$$K(d) = e^{-(d/k)^2}$$

- Can also incorporate nearest neighbors



Feature Scaling

- Warp the input space by scaling features in distance function $D(x,y)=\sqrt{\sum_f w_f d_f(x,y)^2}$
- Larger weight, feature is more relevant



Parameter Selection

- What configuration should be used for prediction?
  - Number of nearest neighbors
  - Kernel width
  - Feature weights
- Search to find the best
- Search Techniques
  - Genetic algorithm
  - Simulated annealing
  - Hill climbing
  - Evaluate a configuration using trace data
- Genetic algorithm tends to work the best
- Not yet satisfied by search performance

Execution Prediction Experiments

- Use IBL techniques just described
  - Limit experience base to 2000 entries
- Predict actual run time / requested run time
  - Improved accuracy a little bit
- Genetic algorithm search for configuration
  - Searched over 1 month of data from steger
- Evaluate using 6 months of data from steger, hopper, lomax (1/01-6/01)

Execution Prediction Performance

Machine	IBL Prediction		Requested Time Prediction		Mean Run Time (minutes)
	Mean Error (minutes)	Percentage of Mean Run Time	Mean Error (minutes)	Percentage of Mean Run Time	
Steger	30.31	32.81	78.00	84.43	92.38
Hopper	16.95	44.37	103.36	270.58	38.20
Lomax	23.00	46.06	126.25	252.85	49.93

Queue Prediction

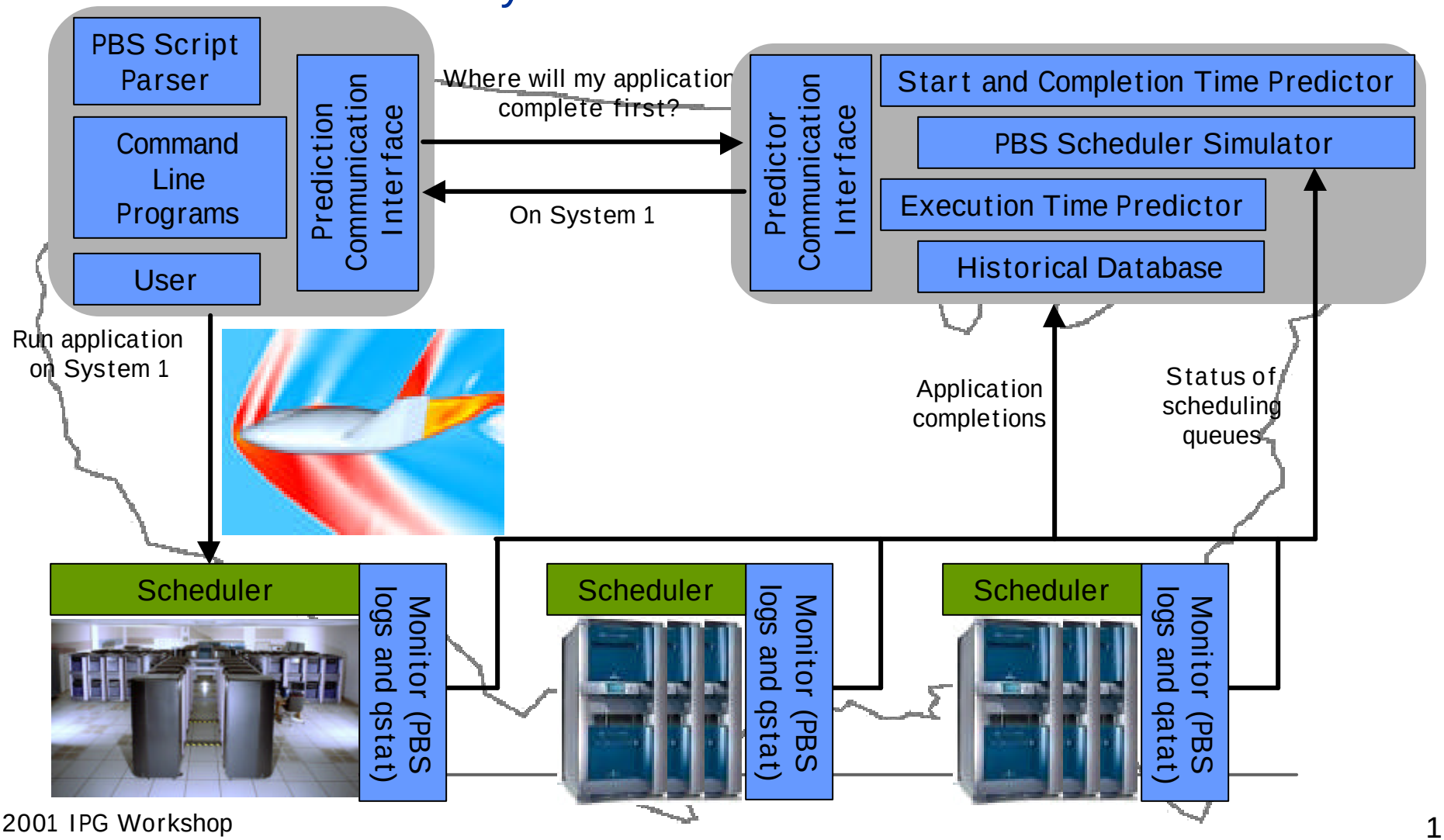
- Predict when a scheduler will start and finish jobs using scheduler simulation
- No simulation mode for PBS
- Wrote our own
  - Event-driven simulator
  - Examine PBS scheduling code
- Use execution time predictions in simulation
- Start time predictions are the simulated start times
- End time predictions are the simulated end times
- Confidence intervals derived by observing past start time prediction error (soon)

Scheduler Simulation Performance

- For 1/01-6/01 on steger:
  - 19777 jobs
  - 12738 (64.41%) matched the actual start times
  - Mismatches are because of dedicated time and crashes
- Haven't had time to evaluate start time prediction accuracy

Implementation

- Predict for 3 Origins at NAS
- From any machine in that cluster



Execution Prediction Implementation

- Separate experience base for each machine
- Used NPBs to compute scaling factors between machines
- Picked between prediction made from the experience base for the machine and a prediction scaled from another machine
  - Picked using size of confidence intervals
- Cache execution predictions to improve response time

Commands

- qruntime
  - Predict how long an application will run on a machine
  - Job already in a queue
  - PBS script with a target machine and queue
- qstarttime
  - Predict when an application will start
- qendtime
  - Predict when an application will finish
- qsuggest
  - Suggest which machine to use

Summary

- Developed techniques to predict application execution times
  - Instance based learning
  - Average error is 33% of average run time
- Developed techniques to predict queue wait times
  - Simulation of scheduling algorithms
  - Execution time predictions
- Implemented these techniques for the NAS Origin cluster
  - Commands to request predictions

Future Work I

- Investigate more advanced instance based learning techniques
- Improve performance of searches
- Extend to predict resource usage (multi-resource scheduling)
- Deploy permanently at NAS
- Integrate into PBS or other schedulers
  - Improve scheduling efficiency
  - Provide predictions to users
- Extend for use in computational grids
  - New architecture
  - Predict time to stage files

Future Work II

- Identify important features (in PBS scripts) to improve prediction performance
  - Number of grid points, number of time steps, …
  - Done by user or tool
- NPB results:
  - 2 runs of class A, B, C NPBs on lomax, steger, hopper
  - 2/3 in the experience base and predicting remaining 1/3:
  - Average run time is 24.08 minutes
  - Error when using requested run time is 13.72 minutes
  - Only NumCPUs, RequestedTime, MachineName:
    error is 4.15 minutes
  - With JobName of <benchmark>-<class>-<# cpus>-<machine>:
    error is 3.33 minutes
  - With Benchmark and Class instead of JobName:
    error is 2.31 minutes